IB/05/050614

Europäisches Patentamt
European Patent Office
Office européen des brevets

Bescheinigung Certificate Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. Patent application No. Demande de brevet n°

04100873.1

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

Anmeldung Nr:
Application no.: 04100873.1
Demande no:

Anmeldetag:
Date of filing: 04.03.04
Date de dépôt:

Anmelder/Applicant(s)/Demandeur(s):

Philips Intellectual Property & Standards
GmbH
Steindamm 94
20099 Hamburg
ALLEMAGNE
Koninklijke Philips Electronics N.V.
Groenewoudseweg 1
5621 BA Eindhoven
PAYS-BAS

Bezeichnung der Erfindung/Title of the invention/Titre de l'invention:
(Falls die Bezeichnung der Erfindung nicht angegeben ist, siehe Beschreibung.
If no title is shown please refer to the description.
Si aucun titre n'est indiqué se referer à la description.)

Verfahren zum Potenzieren bzw. Multiplizieren von Elementen

In Anspruch genommene Priorität(en) / Priority(ies) claimed /Priorité(s) revendiquée(s)
Staat/Tag/Aktenzeichen/State/Date/File no./Pays/Date/Numéro de dépôt:

Internationale Patentklassifikation/International Patent Classification/
Classification internationale des brevets:

G06F7/00

Am Anmeldetag benannte Vertragstaaten/Contracting states designated at date of filing/Etats contractants désignées lors du dépôt:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LU MC NL
PL PT RO SE SI SK TR LI

BESCHREIBUNG

Verfahren zum Potenzieren bzw. Multiplizieren von Elementen

Die vorliegende Erfindung betrifft ein Verfahren zur Multipotenzierung $\Pi_{i=1}^{d} g_i^{e_i}$ bzw. zur Multiskalarmultiplikation $\Sigma_{i=1}^{d} e_i g_i$ von Elementen g_i mittels jeweils mindestens eines jeweils eine maximale Bitanzahl n oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars e_i, insbesondere zur Potenzierung g^e bzw. zur Skalarmultiplikation $e \cdot g$ eines Elements g mittels mindestens eines eine maximale Bitanzahl n oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars e, welche Elemente g_i; g mindestens einer

- im Falle der (Multi-)Potenzierung insbesondere multiplikativ bzw.
- im Falle der (Multi-)Skalarmultiplikation insbesondere additiv

notierten, zum Beispiel Abelschen Gruppe G entstammen.

Bei asymmetrischen Verschlüsselungsverfahren oder Public-Key-Kryptosystemen, die auf der Unlösbarkeit des diskreten Logarithmusproblems in Abelschen Gruppen beruhen, stellt die Potenzierung g^n eines Gruppenelements g oder die Multipotenzierung $g_1^{n_1} \cdot h_k^{n_k}$ mehrerer Gruppenelemente g, h eine der fundamentalen Operationen in Signatur- und Schlüsselaustauschverfahren dar. Die Beschleunigung dieser fundamentalen Operation ist daher von besonderer Bedeutung.

Die Möglichkeit, Potenzen des Gruppenelements g vorzuberechnen, unterliegt dem Problem, dass hierbei das benutzte Gruppenelement g vorher bekannt sein muss. Dies ist etwa im Falle der Signaturverifizierung im D[igital]S[ignature]A[lgorithm] oder im E[lliptic]C[urve] D[igital]S[ignature]A[lgorithm] oder im Diffie-Hellman-Schlüsselaustauschverfahren nicht der Fall. Hinzu kommt, dass zum Beispiel auf Smart-Cards nicht unbedingt genügend Speicherplatz vorhanden ist, um eine hinreichend grosse Anzahl von vorberechneten Elementen abzuspeichern.

Eine andere Möglichkeit besteht in der Rekodierung des benutzten Exponenten; diese Möglichkeit ist unabhängig von der Wahl des Gruppenelements g und daher besonders attraktiv für die Beschleunigung der vorstehend erwähnten Signatur- und Schlüsselaustauschverfahren.

Die Techniken zum Rekodieren des benutzten Exponenten in Algorithmen für (Multi-)Potenzierung beruhen auf der Grundidee, dass eine ganze Zahl neu in einer anderen Form als der üblichen binären Darstellung geschrieben wird, und zwar mit einer kleineren Dichte und mit Koeffizienten in einer endlichen Menge ganzer Zahlen C, die zumindest die Elemente 0 und 1 enthält.

Wenn in der konkreten Gruppe, in der gerechnet wird, das Invertieren eines Elements "gratis" ist, das heißt wenn der rechnerische Aufwand für das Invertieren im Vergleich zu den anderen Gruppenoperationen sehr gering ist und wenn der Gebrauch von vorzeichenbehafteten Koeffizienten gestattet ist, dann kann stets angenommen werden, dass $c \in C$ auch $-c \in C$ impliziert. Wenn das Invertieren rechnerisch aufwendig ist, sind alle Elemente der Menge C nichtnegative ganze Zahlen.

Ein sogenannter "square-and-multiply"-Potenzierungsalgorithmus für die Berechnung von g^e, wobei g ein Gruppenelement und e eine ganze Zahl ist, arbeitet dann bekanntermaßen wie folgt:

- e wird als $\Sigma_{i=0}^{n} e_i 2^i$ geschrieben, wobei jeder Koeffizient e_i in C liegt;
- die Elemente g^{e_n} sind entweder gegeben oder werden vorberechnet;
- die temporäre Variable x wird auf g^{e_n} gesetzt;
- für alle $i = n\text{-}1, n\text{-}2, \ldots, 0$ wird zuerst x quadriert und dann, falls der Koeffizient e_i nichtverschwindend ist, mit dem Element g^{e_i} multipliziert;
- nach der für $i = 0$ erfolgten letzten Quadrierung und gegebenenfalls (nämlich falls Koeffizient e_0 nichtverschwindend ist) nach der Multiplikation mit dem Element g^{e_0} ist der Wert der temporären Variable x das gewünschte Resultat g^e.

Die Anzahl der Gruppenoperationen ist dann etwa gleich der Anzahl der nichtverschwindenden Koeffizienten e_i in der Darstellung $\Sigma_{i=0}^{n} e_i 2^i$ des Exponenten e (diese Gruppenoperationen sind Multiplikationen mit entweder vorberechneten oder gegebenen Gruppenelementen oder, falls die Inversion von Gruppenelementen schnell ist, mit deren Inversen) zuzüglich

- der Länge n der Darstellung (die entsprechenden, etwa n Operationen sind hier Quadrierungen) sowie
- der Mächtigkeit der Tabelle der Elemente g^c, wobei $c \varepsilon C$ und c ungleich Null ist, oder
- der Hälfte dieser Mächtigkeit, falls die Inversion in der gegebenen Gruppe schnell ist und die Koeffizienten e_i vorzeichenbehaftet sind.

Eine gute Abgleichung zwischen der Größe von C und der Dichte der Darstellung ist in der Darstellung des Exponenten der Weg zur optimalen Leistung.

Beispiele von Exponentenrekodierung sind:

- die N[on]A[djacent]F[orm] (vgl. G. W. Reitwiesner, "Binary arithmetic", Advances in Computers 1, Seiten 231 bis 308, 1960; S. Arno und F. S. Wheeler, "Signed digit representations of minimal Hamming weight", IEEE Transactions on Computers 42, 1993, Seiten 1007 bis 1010);
- die zur N[on]A[djacent]F[orm] ähnliche Methode gleichen Gewichts (vgl. M. Joye und S.-M. Yen, "Optimal left-to-right binary signed-digit recoding", IEEE Transactions on Computers 49 (7), 2000, Seiten 740 bis 748);
- die Rekodierung zur Potenzierung mit festen Fenstern (vgl. J. Bos und M. Coster, "Addition chain heuristics", in Advances in Cryptology - CRYPTO '89, LNCS 435, 1990, Seiten 400 bis 407; A. Menezes, P. van Oorschot und S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996);
- die G[eneralized]N[on]A[djacent]F[orm] (vgl. W. E. Clark und J. J. Liang, "On arithmetic weight for a general radix representation ot integers", IEEE Transactions on Information Theory IT-19, 1973, Seiten 823 bis 826);

- "sliding windows" (vgl. E. G. Thurber, "On addition chains $l(mn) \leq l(n)b$ and lower bounds for c(r)", Duke Mathematical Journal 40, 1973, Seiten 907 bis 913; A. Menezes, P. van Oorschot und S. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996), eventuell auf der N[on]A[djacent]F[orm] oder auf weiteren redundanten Basis 2-Darstellungen (vgl. R. Avanzi, "On the complexity of certain multi-exponentiation techniques in cryptography", erscheint in Journal of Cryptology; K. Koyama und Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method", in E. Brickell (Hrsg.), "Advances in Cryptology, Proceedings of Crypto '92", Lecture Notes in Computer Science Band 740, Seiten 345 bis 357, Springer-Verlag, 1992; vgl. auch K. Koyama, Y. Tsuruoka, "A Signed Binary Window Method for Fast Computing over Elliptic Curves", IEICE Trans. Fundamentals, Band E76-A, Nr. 1, Seiten 55 bis 62, Januar 1993); und die
- w[indow]N[on]A[djacent]F[orm] (vgl. J. A. Solinas, "An improved algorithm for arithmetic on a family of elliptic curves", in Advances in Cryptology - CRYPTO '97, B. S. Kaliski jr. (Hrsg.), Lecture Notes in Computer Science Band 1294, Seiten 357 bis 371; H. Cohen, "Analysis of the flexible window powering algorithm", Vorabdruck erhältlich unter http://www.math.u-bordeaux.fr/~cohen/).

Im Hinblick auf die Exponentenrekodierung ist allerdings zu bedenken ist, dass dieses Rekodieren in vielen Fällen nicht "online", das heißt während der Potenzierung selbst erfolgen kann; aus diesem Grunde müssen die rekodierten Exponenten zuerst gespeichert werden. Dieses Erfordernis des Abspeicherns ist jedoch insbesondere in extrem eingeschränkten Umgebungen, wie etwa bei Smart-Cards, nachteilig, denn in einer derartigen, extrem eingeschränkten Umgebung ist jedes Byte des Speichers "kostbar".

Ausgehend von den vorstehend dargelegten Nachteilen und Unzulänglichkeiten sowie unter Würdigung des umrissenen Standes der Technik liegt der vorliegenden Erfindung

die Aufgabe zugrunde, ein Verfahren der eingangs genannten Art so weiterzubilden, dass auch und gerade in extrem eingeschränkten Umgebungen, wie etwa bei Smart-Cards, der Bedarf an Speicherplatz für rekodierte Exponenten bzw. Skalare so stark wie möglich verringert wird.

Diese Aufgabe wird durch ein Verfahren mit den im Anspruch 1 angegebenen Merkmalen gelöst. Vorteilhafte Ausgestaltungen und zweckmäßige Weiterbildungen der vorliegenden Erfindung sind in den Unteransprüchen gekennzeichnet.

Mithin basiert die vorliegende Erfindung auf dem Prinzip des almost-online-Rekodierens für einfaches Exponenzieren bzw. einfaches Skalarmultiplizieren oder für Multipotenzieren bzw. Multiskalarmultiplizieren in beschränkten rechnerischen Umgebungen; in diesem Zusammenhang bedeutet "almost-online"-Rekodieren, dass der Exponent bzw. Skalar in Abschnitte gespalten wird, die einzeln rekodiert werden und deren Rekodierung schichtweise zwischen Teilen der (Multi-)Potenzierung bzw. der (Multi-)Skalarmultiplikation erfolgt.

Die Technik der "almost-online"-Rekodierung kann zur Verringerung des Speicherbedarfs für die rekodierten Exponenten bzw. für die rekodierten Skalare eingesetzt werden. Die Auswirkungen der almost-online-Rekodierung auf die Totallaufzeit der (Multi-)Potenzierung bzw. der (Multi-)Skalarmultiplikation sind in der Regel minimal.

Gedanklich ausgehend von den vorstehend erwähnten exemplarischen Rekodierungen, wird im Verfahren gemäß der vorliegenden Erfindung angenommen, dass die Rekodierung im Falle der Multipotenzierung bzw. der Multiskalarmultiplikation von der Form $e_i = \Sigma_{j=0}^{n} b_{i,j} 2^j$ ist; im Falle der (einfachen) Potenzierung bzw. der (einfachen) Skalarmultiplikation, die ein Spezialfall der Multipotenzierung bzw. der Multiskalarmultiplikation ist, wird entsprechend von der Annahme $e = \Sigma_{j=0}^{n} b_j 2^j$ ausgegangen, wobei $n = \lfloor \log_2 e \rfloor$ die Bitlänge von e ist, dass heißt diese Bitlänge n ist höchstens ein Bit länger als die binäre Darstellung. Dies bedeutet mit anderen Worten, dass $n+1$ als die maximale Länge eines jeden Exponenten bzw. Skalars $e_i = \Sigma_{j=0}^{n} b_{i,j} 2^j$ zu verstehen ist.

Des weiteren wird angenommen, dass der rekodierende Algorithmus - möglicherweise nicht explizit - von einem Parameter w abhängt, der normalerweise der Breite eines Fensters, über dem die Bits der Exponenten bzw. Skalare e_i gelesen werden, oder der oberen Schranke einer solchen Breite entspricht.

Auf dieser Grundlage erfolgt gemäß der Lehre der vorliegenden Erfindung die symbolhaft in der Schreibweise $\Pi_{i=1}{}^{d} g_i^{e_i}$ ausdrückbare Multipotenzierung im Falle einer multiplikativ notierten, insbesondere Abelschen Gruppe G in den folgenden Schritten:

zunächst: Wählen einer Brocken- oder Teilbreite L, die wesentlich größer als der Parameter w und wesentlich kürzer als die maximale Länge eines jeden Exponenten e_i sein kann;

sodann:

[a.1] Berechnen und Speichern oder

[a.2] Wiedererlangen aus einem Speicher
aller Potenzen g_i^c,
wobei g_i ein Element der Gruppe G und
c ein zulässiger positiver Koeffizient ist;

[b] Unterteilen eines jeden, insbesondere ganzzahligen Exponenten e_i in mehrere Brocken oder in mehrere Teile $e_{i,k}$ mit der vorstehend gewählten Brocken- oder Teilbreite L,

[b.1] wobei der Exponent e_i in der unterteilten Form $e_i = \Sigma_{k=0}{}^{r} e_{i,k} 2^{kL}$ mit $0 \leq e_{i,k} < 2^L$ geschrieben werden kann und

[b.2] wobei die Anzahl r der Brocken oder Teile $e_{i,k}$ als insbesondere ganzzahliger Quotient aus der maximalen Bitanzahl n und der Bitanzahl L der Brocken- oder Teilbreite definiert werden kann;

[c] einzelnes Rekodieren der Brocken oder Teile $e_{i,k}$, wobei dieses Rekodieren für jeden einzelnen Brocken bzw. für jeden einzelnen Teil $e_{i,k}$ eines jeden Exponenten e_i in folgende Teilschritte unterteilt werden kann:

[c.1] Setzen einer temporären Variable x auf einen normierten Wert, insbesondere auf den Wert 1, wobei mit 1 das neutrale Element der Gruppe G bezüglich der der Gruppe G zugehörigen Gruppenoperation bezeichnet wird;

[c.2] Setzen einer Variable k auf die Werte r-1, r-2, ..., 0 (einer nach dem anderen), wobei für jeden derartigen Wert $k = r$-1, r-2, ..., 0 der Variable k die folgenden Teilschritte ausgeführt werden:

[c.2.i] für jeden Wert $i = 1, 2, ..., d$ eines Index i, wobei d als Anzahl der Elemente g_i und der den Elementen g_i zugehörigen Exponenten e_i definiert wird:

[c.2.i.a] Rekodieren des Brockens oder Teils $e_{i,k}$ als Summe $\Sigma_{j=0}{}^{L} b_{i,j}2^j$ aus mit jeweils einem einer endlichen Menge C ganzer Zahlen entstammenden Koeffizienten $b_{i,j}$ gewichteten Zweierpotenzen 2^j;

[c.2.i.b] wenn der der höchsten Zweierpotenz 2^L zugeordnete Koeffizient $b_{i,L}$ nicht verschwindet: Setzen der temporären Variable x auf das Produkt von x und der dem Koeffizienten $b_{i,L}$ der höchsten Zweierpotenz 2^L zugeordneten Potenz $g_i^{b_{i,L}}$ des Elements g_i;

[c.2.ii] für jeden Wert $j = L$-1, L-2, ..., 0 des Index j:

[c.2.ii.a] Quadrieren der temporären Variable x;

[c.2.ii.b] für jeden Wert $i = 1, 2, ..., d$ des Index i:

wenn der der Zweierpotenz 2^j zugeordnete Koeffizient $b_{i,j}$ nicht verschwindet: Setzen der temporären Variable x auf das Produkt von x und der dem Koeffizienten $b_{i,j}$ der Zweierpotenz 2^j zugeordneten Potenz $g_i^{b_{i,j}}$ des Elements g_i;

schließlich: Wiedergabe von x.

Der Spezialfall der (einfachen) Potenzierung ergibt sich vorstehend für $d = 1$, das heißt bei Vorliegen eines einzigen Elements g sowie eines einzigen dem Element g zugehörigen Exponenten e, was de facto mit einem Weglassen des Index i gleichzusetzen ist; in diesem Falle wird dann also ein Element g mittels eines eine maximale Bitanzahl n oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten e zu einer Potenz g^e potenziert, wobei das Element g wiederum einer multiplikativ notierten Abelschen Gruppe G entstammt.

In analoger Weise erfolgt gemäß der Lehre der vorliegenden Erfindung die symbolhaft in der Schreibweise $\Sigma_{i=1}{}^{d} e_i g_i$ ausdrückbare Multiskalarmultiplikation im Falle einer additiv notierten, insbesondere Abelschen Gruppe G in den folgenden Schritten:

zunächst: Wählen einer Brocken- oder Teilbreite L, die wesentlich größer als der Parameter w und wesentlich kürzer als die maximale Länge eines jeden Skalars e_i sein kann;

sodann:

[a.1] Berechnen und Speichern oder

[a.2] Wiedererlangen aus einem Speicher
aller Vielfachen $c \cdot g_i$,
wobei c ein zulässiger positiver Koeffizient und
g_i ein Element der Gruppe G ist;

[b] Unterteilen eines jeden, insbesondere ganzzahligen Skalars e_i in mehrere Brocken oder in mehrere Teile $e_{i,k}$ mit der vorstehend gewählten Brocken- oder Teilbreite L,

[b.1] wobei der Skalar e_i in der unterteilten Form $e_i = \Sigma_{k=0}{}^{r} e_{i,k} 2^{kL}$ mit $0 \leq e_{i,k} < 2^L$ geschrieben werden kann und

[b.2] wobei die Anzahl r der Brocken oder Teile $e_{i,k}$ als insbesondere ganzzahliger Quotient aus der maximalen Bitanzahl n und der Bitanzahl L der Brocken- oder Teilbreite definiert werden kann;

[c] einzelnes Rekodieren der Brocken oder Teile $e_{i,k}$, wobei dieses Rekodieren für jeden einzelnen Brocken bzw. für jeden einzelnen Teil $e_{i,k}$ eines jeden Skalars e_i in folgende Teilschritte unterteilt werden kann:

[c.1] Setzen einer temporären Variable x auf einen normierten Wert, insbesondere auf den Wert 0, wobei mit 0 das neutrale Element der Gruppe G bezüglich der der Gruppe G zugehörigen Gruppenoperation bezeichnet wird;

[c.2] Setzen einer Variable k auf die Werte r-1, r-2, ..., 0 (einer nach dem anderen), wobei für jeden derartigen Wert $k = r$-1, r-2, ..., 0 der Variable k die folgenden Teilschritte ausgeführt werden:

[c.2.i] für jeden Wert $i = 1, 2, ..., d$ eines Index i, wobei d als Anzahl der Elemente g_i und der den Elementen g_i zugehörigen Skalare e_i definiert wird:

[c.2.i.a] Rekodieren des Brockens oder Teils $e_{i,k}$ als Summe $\Sigma_{j=0}^{L} b_{i,j}2^j$ aus mit jeweils einem einer endlichen Menge C ganzer Zahlen entstammenden Koeffizienten $b_{i,j}$ gewichteten Zweierpotenzen 2^j;

[c.2.i.b] wenn der der höchsten Zweierpotenz 2^L zugeordnete Koeffizient $b_{i,L}$ nicht verschwindet: Setzen der temporären Variable x auf die Summe von x und des dem Koeffizienten $b_{i,L}$ der höchsten Zweierpotenz 2^L zugeordneten Vielfachen $b_{i,L} \cdot g_i$ des Elements g_i;

[c.2.ii] für jeden Wert $j = L\text{-}1, L\text{-}2, ..., 0$ des Index j:

[c.2.ii.a] Verdoppeln der temporären Variable x;

[c.2.ii.b] für jeden Wert $i = 1, 2, ..., d$ des Index i:

wenn der der Zweierpotenz 2^j zugeordnete Koeffizient $b_{i,j}$ nicht verschwindet: Setzen der temporären Variable x auf die Summe von x und des dem Koeffizienten $b_{i,j}$ der Zweierpotenz 2^j zugeordneten Vielfachen $b_{i,j} \cdot g_i$ des Elements g_i;

schließlich: Wiedergabe von x.

Der Spezialfall der (einfachen) Skalarmultiplikation ergibt sich vorstehend für $d = 1$, das heißt bei Vorliegen eines einzigen Elements g sowie eines einzigen dem Element g zugehörigen Skalars e, was de facto mit einem Weglassen des Index i gleichzusetzen ist; in diesem Falle wird dann also ein Element g mit einem eine maximale Bitanzahl n oder Bitlänge aufweisenden, insbesondere ganzzahligen Skalar e zu einem Produkt $e \cdot g$ multipliziert, wobei das Element g wiederum einer additiv notierten Abelschen Gruppe G entstammt.

Gemäß einer bevorzugten Weiterbildung der vorliegenden Erfindung wird

- der rekodierte Brocken oder der rekodierte Teil $e_{i,k}$ einmal verwendet und
- die Speichereinheit, in der der rekodierte Brocken oder der rekodierte Teil $e_{i,k}$ gespeichert wird, für das Rekodieren des folgenden Brockens oder des folgenden Teils $e_{i,k-1}$ eingesetzt,

so dass der Speicherbedarf von auf rechts-nach-links-Rekodierungen ganzer Zahlen basierenden (Multi-)Potenzierungsalgorithmen bzw. (Multi-)Skalarmultiplikationsalgorithmen beträchtlich verringert werden kann.

Die vorliegende Erfindung betrifft des weiteren einen Mikroprozessor, arbeitend gemäß einem Verfahren gemäß der vorstehend dargelegten Art.

Die vorliegende Erfindung betrifft des weiteren eine Vorrichtung, insbesondere Chipkarte und/oder insbesondere Smart-Card, aufweisend mindestens einen Mikroprozessor gemäß der vorstehend dargelegten Art.

Die vorliegende Erfindung betrifft schließlich die Verwendung

- eines Verfahrens gemäß der vorstehend dargelegten Art und/oder
- mindestens eines Mikroprozessors gemäß der vorstehend dargelegten Art und/oder
- mindestens einer Vorrichtung, insbesondere mindestens einer Chipkarte und/oder insbesondere mindestens einer Smart-Card, gemäß der vorstehend dargelegten Art

bei mindestens einem Kryptosystem, insbesondere bei mindestens einem Public-Key-Kryptosystem, bei mindestens einem Schlüsselaustauschsystem oder bei mindestens einem Signatursystem.

Wie bereits vorstehend erörtert, gibt es verschiedene Möglichkeiten, die Lehre der vorliegenden Erfindung in vorteilhafter Weise auszugestalten und weiterzubilden. Hierzu wird einerseits auf die dem Anspruch 1 nachgeordneten Ansprüche verwiesen, andererseits werden weitere Ausgestaltungen, Merkmale und Vorteile der vorliegenden Erfindung nachstehend anhand der exemplarischen Implementierung von fünf Ausführungsbeispielen näher erläutert, wobei sich

- das erste Ausführungsbeispiel auf das Verfahren der Einfachpotenzierung (= einfache Exponenzierung),

- das zweite Ausführungsbeispiel auf das Verfahren der Multipotenzierung und
- das dritte Ausführungsbeispiel ebenfalls auf das Verfahren der Multipotenzierung

beziehen, das heißt von einer multiplikativen Notation für die Abelsche Gruppe G ausgehen und wobei sich

- das vierte Ausführungsbeispiel auf das Verfahren der Einfachskalarmultiplikation (= einfache Skalarmultiplikation) und
- das fünfte Ausführungsbeispiel auf das Verfahren der Multiskalarmultiplikation

beziehen, das heißt von einer additiven Notation für die Abelsche Gruppe G ausgehen (im Falle einer derartigen additiven Notation für die Abelsche Gruppe G sind im Vergleich zur multiplikativen Notation für die Abelsche Gruppe G im vorstehenden Kapitel "Stand der Technik" offensichtlicherweise Änderungen und Ersetzungen vorzunehmen, die aus den zwischen Anspruch 4 [<--> (Multi-)Potenzierung: neutrales Element "1"; "Quadrieren"; "Produkt"] und Anspruch 5 [<--> (Multi-)Skalarmultiplikation: neutrales Element "0"; "Verdoppeln"; "Summe"] unterschiedlichen Formulierungen ersichtlich sind.

Die nachstehend zur vorliegenden Erfindung dargelegten fünf Ausführungsbeispiele bedienen sich einer allgemeinen Technik in Form der sogenannten almost-online-Rekodierung, die eingesetzt werden kann, um den Speicherbedarf von auf rechts-nach-links-Rekodierungen ganzer Zahlen basierenden

- Einfachpotenzierungsalgorithmen (vgl. erstes Ausführungsbeispiel),
- Multipotenzierungsalgorithmen (vgl. zweites Ausführungsbeispiel und drittes Ausführungsbeispiel),
- Einfachskalarmultiplizierungsalgorithmen (vgl. viertes Ausführungsbeispiel) oder
- Multiskalarmultiplizierungsalgorithmen (vgl. fünftes Ausführungsbeispiel)

beträchtlich zu verringern.

Die Technik der almost-online-Rekodierung kann in extrem beschränkten rechnerischen Umgebungen, wie zum Beispiel in Chipkarten oder in SmartCards, sehr nützlich sein, wobei die Ersparnis an Speicherplatz in Abhängigkeit von der bestimmten Situation

beträchtlich sein kann (möglicherweise tritt ein allerdings sehr geringer Durchsatzverlust auf, und zwar insbesondere dann, wenn der Exponent bzw. Skalar in zu viele kleine Teile (= in zu viele kleine "Brocken") unterteilt wird; dann kann die Auswirkung auf die Leistung durchaus sprübar sein).

Erstes Ausführungsbeispiel: einfache Exponenzierung = "single exponentiation"

Wenn G eine Abelsche Gruppe mit einer Ordnung der Größenordnung 2^n ist und angenommen wird, dass ein Element $g \in G$ und eine ganze Zahl e gegeben sind, so wird erfindungsgemäß darauf abgezielt, $x = g^e$ schnellstmöglich zu berechnen. Die erfindungsgemäß ausgesuchte Rekodierung macht die Potenzierung sehr schnell, jedoch kann diese Rekodierung nicht online benutzt werden, das heißt nicht während der Potenzierung selbst erfolgen; dieses ist zum Beispiel bei der w[indow]N[on]A[djacent] F[orm] der Fall.

Die beim almost-online-Rekodieren eingesetzte Technik besteht nun darin, den Exponenten e in mehrere "Exponentenbrocken", das heißt in mehrere Exponentenabschnitte oder in mehrere Exponententeile zu unterteilen, die beträchtlich länger als w Bits, aber gleichwohl viel kürzer als e sind. Die Brocken oder Teile werden dann einzeln rekodiert, einmal verwendet, und danach wird der Speicher, in dem die Brocken oder Teile gespeichert worden sind, für das Rekodieren des folgenden Brockens bzw. des folgenden Teils wiederverwendet, so dass der für den Exponenten n insgesamt benötigte Speicherplatz in signifikanter Weise verringert werden kann.

Die nachstehende almost-online-Rekodierung erfolgt unter der Annahme, dass die Brocken oder Teile eine Länge von L Bits aufweisen. Der Grund dafür, dass L wesentlich größer als w ist, besteht darin, dass die Abschätzungen für die Anzahl nichtverschwindender Koeffizienten in rekodierten Exponenten normalerweise asymptotisch gegeben werden, die tatsächliche Anzahl nichtverschwindender Koeffizienten in rekodierten Exponenten jedoch manchmal infolge einer kleinen additiven Konstante größer ist, was anhand eines konkreten Beispiels weiter unten noch veranschaulicht wird.

Nachfolgend wird im Rahmen des ersten Ausführungsbeispiels der almost-online-Rekodierung ein Algorithmus vorgestellt, bei dem

- ein Basiselement g der Abelschen Gruppe G,
- eine ganze Zahl e mit n Bits,
- eine Fensterbreite w und
- eine Brocken- oder Teilbreite $L >> w$

eingegeben werden; die (Einfach-)Potenzierung g^e wird ausgegeben:

Schritt 1. $x \leftarrow 1$

Schritt 2. $r \leftarrow \lceil n/L \rceil$, dann betrachte $e = \sum_{k=0}^{r-1} e_k 2^{kL}$ mit $0 \leq e_k < 2^L$

Schritt 3. **for** $k = r-1$ **downto** 0 **do** {

(a) `Rekodiere`$(e_k) \rightarrow e_k = \sum_{j=0}^{L} b_j 2^j$

(b) **if** $b_L \neq 0$ **then** $x \leftarrow x \cdot g^{b_L}$

(c) **for** $j = L-1$ **downto** 0 **do** {

(i) $x \leftarrow x^2$

(ii) $x \leftarrow x \cdot g^{b_j}$ } }

Schritt 4. `return` x

Hierzu ist anzumerken, dass es nach jeweils L Bits passieren kann, dass der vorstehende Algorithmus zwei Gruppenmultiplikationen in einer Reihe anstelle von nur einer Gruppenmultiplikation durchführt. Dies geschieht, wenn einer der Brocken e_i (= einer der Exponententeile e_i) eine ungerade Zahl darstellt und wenn das Rekodieren des folgenden Brockens e_{i+1} (= des folgenden Exponententeils e_{i+1}) um einen Koeffizienten länger ist (b_L ungleich Null).

Es kann nun mittels eines konkreten Beispiels, bei dem die gewählte Rekodierung die w[indow]N[on]A[djacent]F[orm] ist, gezeigt werden, dass der Geschwindigkeitsverlust minimal ist und dass die Speicherersparnis ziemlich groß sein kann:

Für $n = 160$ ist der optimale Wert von w gleich 5 (vgl. H. Cohen, "Analysis of the flexible window powering algorithm", Vorabdruck erhältlich unter http://www.math.u-bordeaux.fr/~cohen/); damit sind sieben Potenzen $g^3, g^5, g^7, g^9, g^{11}, g^{13}, g^{15}$ des Basiselements g vorzuberechnen, und g^2 wird auch vorübergehend benötigt. Mindestens fünf Bits pro rekodiertem Koeffizienten sind erforderlich, aber der Implementor verwendet vermutlich vollständige vorzeichenbehaftete Bytes.

Zwei rekodierte Exponenten erfordern 320 Bytes R[andom]A[ccess]M[emory], aber zwei rekodierte 32 Bit-Brocken (= 32 Bit-Abschnitte oder 32 Bit-Teile) erfordern nur 66 Bytes R[andom]A[ccess]M[emory]. Die eingesparten 254 Bytes R[andom]A[ccess] M[emory] können verwendet werden, um sechs Punkte einer elliptischen Kurve in affinen Koordinaten zu speichern.

Cohen hat nun bewiesen (vgl. H. Cohen, "Analysis of the flexible window powering algorithm", Vorabdruck erhältlich unter http://www.math.u-bordeaux.fr/~cohen/), dass das durchschnittliche Hamming-Gewicht der w[indow]N[on]A[djacent]F[orm] einer ganzen Zahl mit n Bits (, was die durchschnittliche Anzahl der Multiplikationen in der entsprechenden Potenzierung plus einer ist,) gleich

$$n/(w+1) + 1 - 0{,}5(w-1)(w+2)/(w+1)^2 + O(\rho^{-n})$$

ist, wobei $\rho = \rho(w)$ eine reelle Zahl größer als eins ist, die nur von w und nicht von n abhängig ist. Numerisch ist $\rho = 2^{1/2} = 1{,}414...$ für $w = 3$,

$\rho = 1{,}2157...$ für $w = 4$ und

$\rho = 1{,}1296...$ für $w = 5$.

Der vorstehende Satz hinsichtlich des durchschnittlichen Hamming-Gewichts der w[indow]N[on]A[djacent]F[orm] impliziert, dass, wenn eine ganze Zahl in r Brocken bzw. in r Teile aufgespaltet wird, das gesamte Hamming-Gewicht der r Brocken bzw. der r Teile um

$$(r-1)(1-0{,}5(w-1)(w+2)/(w+1)^2)$$

größer als das Hamming-Gewicht der ursprünglichen ganzen Zahl ist.

Im Fall $n = 160$ kann $L = 32$ und folglich $r = 5$ gewählt werden. Die "flexible window"-Methode erfordert durchschnittlich 22/9 = 2,44 weniger Gruppenoperationen als die almost-online-Methode gemäß der vorliegenden Erfindung. Dieser Unterschied beträgt etwa 1,26 Prozent der Gesamtlaufzeit des Potenzierungsalgorithmus (über die 193 Gruppenoperationen, einschließlich der Zeit für die Vorberechnungen); jedoch ist der Speicherbedarf für die rekodierten Exponenten um etwa achtzig Prozent verringert worden.

Zweites Ausführungsbeispiel: Multipotenzierung

Der vorstehende Algorithmus aus dem ersten Ausführungsbeispiel (einfache Exponenzierung = "single exponentiation") kann in ein Multipotenzierungsverfahren transformiert werden.

Wenn Gruppenelemente $g_1, ..., g_d \varepsilon G$ und Exponenten $e_1, ..., e_d$ mit $d > 1$ gegeben sind und $\Pi_{i=1}^{d} g_i^{e_i}$ berechnet werden soll, wird zunächst eine Entscheidung für eine Verwendung einer dünnbesetzten Rekodierung der Exponenten $e_1, ..., e_d$ getroffen; dann wird eine "square-and-multiply"-Schleife verwendet:

Zuerst werden alle Potenzen g_i^c berechnet und gespeichert, wobei c ein zulässiger positiver Koeffizient ist. Dann wird eine temporäre Variable x auf $1 \varepsilon G$ gesetzt. Für $j = n, n-1, ..., 0$ wird zuerst x quadriert, und für $i = 1, ..., d$ wird das quadrierte x mit $g_i^{e_{i,j}}$ multipliziert, wobei $e_{i,j}$ der Koeffizient von 2^j in der Rekodierung von e_i ist. Am Ende enthält die temporäre Variable x das gewünschte Resultat.

Diese Methode wird auch als durchgeschobene Potenzierung bezeichnet; wie schon bei der Situation gemäß dem ersten Ausführungsbeispiel ist es wiederum wünschenswert, die Vorteile einer guten rechts-nach-links-Rekodierung beizubehalten, ohne zu viel Speicher verwenden zu müssen.

Die folgende Variante führt die Rekodierung "almost online", das heißt nahezu während der durchgeschobenen Multipotenzierung oder kurz nach der durchgeschobenen Multipotenzierung durch, wobei in den Algorithmus

- Basiselemente $g_1, ..., g_d$ der Abelschen Gruppe G,
- ganze Zahlen $e_1, ..., e_d$ ($d > 1$) mit jeweils höchstens n Bits,
- eine Fensterbreite w,
- eine Brocken- oder Teilbreite $L >> w$ und
- vorberechnete Potenzen g_i^c für alle c in der Koeffizientenmenge

eingegeben werden; die Multipotenzierung oder Vielfachpotenzierung $\Pi_{i=0}{}^{d} g_i^{e_i}$ wird ausgegeben:

Schritt 1. $x \leftarrow 1$

Schritt 2. $r \leftarrow \lceil n/L \rceil$, dann betrachte $e_i = \sum_{k=0}^{r-1} e_{i,k} 2^{kL}$ for $i = 1 \ldots d$

Schritt 3. **for** $k = r-1$ **downto** 0 **do** {

(a) **for** $i = 1$ **to** d **do** {

$\texttt{Recode}(e_{i,k}) \rightarrow e_{i,k} = \sum_{j=0}^{L} b_{i,j} 2^j$

if $b_{i,L} \neq 0$ **then** $x \leftarrow x \cdot g_i^{b_{i,L}}$ }

(b) **for** $j = L-1$ **downto** 0 **do** {

(i) $x \leftarrow x^2$

(ii) **for** $i = 1$ **to** d **do** { **if** $b_{i,L} \neq 0$ **then** $x \leftarrow x \cdot g_i^{b_{i,j}}$ }

} }

Schritt 4. **return** x

Die zum Algorithmus gemäß dem ersten Ausführungsbeispiel gemachten Anmerkungen sind auch hier relevant, das heißt im Falle von elliptischen Kurven über einem endlichen Körper mit $n = 160$ und $L = 32$ werden $2{,}44d$ Gruppenoperationen verwendet, wobei d die Anzahl der Potenzen ist, die miteinander multipliziert werden sollen. Dies ist zwar mehr als bei der einfachen durchgeschobenen Potenzierung, jedoch können $254d$ Bytes des R[andom]A[ccess]M[emory]s eingespart werden, das heißt Speicher für $6d$ vorberechnete Punkte in affinen Koordinaten.

Drittes Ausführungsbeispiel:

Multipotenzierung mit parallelen schiebenden Fenstern

Im dritten Ausführungsbeispiel wird der Einsatz der almost-online-Rekodierung in einer Verallgemeinerung (vgl. R. Avanzi, "On the complexity of certain multi-exponentiation techniques in cryptography", erscheint in Journal of Cryptology) eines Algorithmus von Yen, Laih und Lenstra (vgl. S.-M. Yen, C.-S. Laih und A. K. Lenstra, "Multi-exponentiation", IEE Proc. Comput. Digit. Tech., Band 141, Nr. 6, November 1994) beschrieben.

In diesem Zusammenhang dient dieses nachfolgend dargelegte dritte Ausführungsbeispiel vornehmlich der Erläuterung der Grundprinzipien des beschriebenen Algorithmus; die erlangbare Effizienzsteigerung ist als eher gering einzuschätzen. Der Algorithmus ist im wesentlichen eine Variante des Tricks von Shamir mit einem gleitenden Fenster ("sliding window") und wird im folgenden dargestellt:

In den Algorithmus werden

- eine Fensterbreite w,
- ganze Zahlen $e_i = \Sigma_{j=0}^{n} e_{i,j} 2^j$ und
- eine Menge E von vorberechneten Elementen aus der Gruppe G der Form $\Pi_{i=1}^{d} g_i^{k_i}$ einschließlich $g_1, ..., g_d$ (die Menge E ist stark von der Fensterbreite w und von der Darstellung der ganzen Zahlen e_i abhängig; vgl. die Anmerkung nach dem nachstehenden Algorithmus)

eingegeben; die Multipotenzierung oder Vielfachpotenzierung $\Pi_{i=1}^{d} g_i^{e_i}$ wird ausgegeben:

Schritt 1. $t \leftarrow n$ und $x \leftarrow 1 \in G$

Schritt 2. **if** $(e_{i,t-1} = 0$ for $i = 1, 2, \ldots, d)$ **then** {

(a) $t \leftarrow t - 1$ and $x \leftarrow x^2$

} **else** {

(b) **if** $t \geq w$ **then** $t \leftarrow t - w$ **else** { $w \leftarrow t$ and $t \leftarrow 0$ }

(c) **for** $i = 1, 2, \ldots, d$ **do** $f_i \leftarrow \sum_{j=0}^{w-1} e_{i,t+j} 2^j$

(d) Sei s die größte natürliche Zahl $s \geq 0$ derart, dass $2^s | f_i$ für alle i

(e) **for** $i = 1, 2, \ldots, d$ **do** $f_i \leftarrow f_i / 2^s$

(f) (i) $x \leftarrow x^{2^{w-s}}$; (ii) $x \leftarrow x \cdot \prod_{i=1}^{d} g_i^{f_i}$ and (iii) $x \leftarrow x^{2^s}$ }

Schritt 3. **if** $t = 0$ **then return** x **else goto Schritt 2**

Hierzu ist anzumerken, dass f_i zu Beginn von Schritt 2.(c) die durch eine Kette von w aufeinanderfolgenden Bits des Exponenten e_i dargestellte ganze Zahl ist. Nach dem Normalisierungsschritt 2.(e) ist mindestens eine der f_i ungerade.

Wenn in der Gruppe G die Inversion von Elementen schnell erfolgt, wird die N[on]A[djacent]F[orm] als Rekodierung gewählt. Es ist einfach zu sehen, dass die Anzahl von vorzeichenbehafteten ganzen Zahlen mit w Bits in der N[on]A[djacent]F[orm] $I_w = (2^{w+2}-(-1)^w)/3$ beträgt. Die Menge E enthält alle Elemente der Form $\Pi_{i=1}{}^{d} g_i^{k_i}$ derart, dass

- $|k_i| \leq T_w$ für $i = 1, 2, \ldots, d$,
- mindestens eine der k_i ungerade und
- der erste nichtverschwindende Wert aus der Folge $k_1, k_2, \ldots, k_p$ positiv

ist. Auf diese Weise kann Schritt 2.(f)(ii) entweder mit einer Multiplikation oder mit einer Division durchgeführt werden. Die Mächtigkeit von E beträgt $(I_w{}^d - I_{w-1}{}^d)/2$.

Nun werden die Parameter $w = 2 = d$ fixiert und die N[on]A[djacent]F[orm] für die Rekodierung der Exponenten gewählt. Die Motivation hierfür ist die Herstellung von digitalen Signaturen mit elliptischen Kurven (vgl. American National Standards Institute, "ANSI X9.62: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999):

In diesem Falle ist $d = 2$, und für die interessante Größe der Exponenten, nämlich von $n = 160$ bis $n = 240$, ist der Parameter $w = 2$ optimal (vgl. R. Avanzi, "On the complexity of certain multi-exponentiation techniques in cryptography", erscheint in Journal of Cryptology). Der vorstehende Algorithmus aus dem dritten Ausführungsbeispiel wird also zu einer almost-online-Multipotenzierung mit $d = 2 = w$ und der N[on]A[djacent]F[orm], wobei in den Algorithmus

- zwei (Basis-)Elemente g_1, g_2 der Abelschen Gruppe G,
- zwei natürliche Zahlen e_1, e_2 mit jeweils höchstens n Bits und
- eine Brocken- oder Teilbreite L mit $n >> L >> 2$

eingegeben werden; die Zweifachpotenzierung $g_1^{e_1} \cdot g_2^{e_2}$ wird ausgegeben:

Schritt 1. Berechne die 8 Elemente $g_1^a g_2^b$ vor, mit entweder $0 < a \leq 2$ und $-2 \leq b \leq 2$ wobei mindestens eine von a, b ungerade ist, oder $a = 0$ und $b = 1$. [Siehe Remark A.2]

Schritt 2. $x \leftarrow 1$

$r \leftarrow \lceil n/L \rceil$, dann betrachte $e_i = \sum_{k=0}^{r-1} e_{i,k} 2^{kL}$ for $i = 1, 2$ with $0 \leq e_{i,k} < 2^L$

Schritt 3. **for** $k = r - 1$ **downto** 0 **do** {

(a) **for** $i = 1, 2$ **do** Rekodiere $e_{i,k}$ **als** NAF : $\nu_i := e_{i,k} = \sum_{j=0}^{L} \nu_{i,j} 2^j$

$a_1 \leftarrow 0$, $a_2 \leftarrow 0$

(b) **if** $(\nu_{1,L}, \nu_{2,L}) \neq (0,0)$ **then** {

(i) **if** $(\nu_{1,L-1}, \nu_{2,L-1}) = (0,0)$ **then** $x \leftarrow x \cdot (g_1^{\nu_{1,L}} \cdot g_2^{\nu_{2,L}})$

(ii) **else** { $a_1 \leftarrow \nu_{1,L}$, $a_2 \leftarrow \nu_{2,L}$ } }

(c) **for** $j = L - 1$ **downto** 0 **do** {

(i) $x \leftarrow x^2$

(ii) **if** $(\nu_{1,j}, \nu_{2,j}) \neq (0,0)$ **then** {

if $(a_1, a_2) \neq (0,0)$ **then** {

(iii) $a_1 \leftarrow 2a_1 + \nu_{1,j}$, $a_2 \leftarrow 2a_2 + \nu_{2,j}$

(iv) $x \leftarrow x \cdot (g_1^{a_1} \cdot g_2^{a_2})$ (or: $x \leftarrow x/(g_1^{-a_1} \cdot g_2^{-a_2})$)

} **else** {

if ($j > 0$ **and** $(\nu_{1,j-1}, \nu_{2,j-1}) \neq (0,0)$) **then** {

(v) $a_1 \leftarrow \nu_{1,j}$, $a_2 \leftarrow \nu_{2,j}$

} **else** {

(vi) $x \leftarrow x \cdot (g_1^{\nu_{1,j}} \cdot g_2^{\nu_{2,j}})$ (or: $x \leftarrow x/(g_1^{-\nu_{1,j}} \cdot g_2^{-\nu_{2,j}})$)

} } }

} *(End der inneren **for**-Schleife)*

} *(End der äusseren **for**-Schleife)*

Schritt 4. `return` x

Hierzu ist anzumerken, dass in Schritt 3 die beiden verschachtelten Schleifen des vorstehenden Algorithmus aus dem ersten Ausführungsbeispiel und das gleichzeitige sequentielle Abfragen des vorstehenden ersten Algorithmus aus dem dritten Ausführungsbeispiel zu erkennen sind.

In den Schritten 3.(c)(ii), 3.(c)(iii), 3.(c)(iv), 3.(c)(v), 3.(c)(vi) werden Fenster der Breite 2 über die gekoppelten N[on]A[djacent]F[orm]s von zwei Brocken bzw. von zwei Teilen mit L Bits zusammengestellt.

Zwei "Überträge" a_1 und a_2 speichern die Werte einer nichtverschwindenden Spalte, falls die folgende Spalte auch nichtverschwindend ist, so dass die Werte während der nächsten Iteration verdoppelt und zu den Werten in der nächsten Spalte addiert werden können; vgl. Schritt 3.(c)(iii) . Die Schritte 3.(c)(iv) und 3.(c)(vi) werden mit einer Multiplikation oder mit einer Division durchgeführt.

Wenn nun zwei ganze Zahlen b_1 und b_2 als $b_i = \Sigma_{t=1}^{m} b_{i,j}2^j$ geschrieben werden, besteht eine Spalte aus einem Paar von Koeffizienten $(b_{1,t}, b_{2,t})$ aus den vorstehenden Darstellungen. Die geordnete Folge derartiger Spalten ist die gemeinsame Darstellung von b_1 und b_2. Die Anzahl von nichtverschwindenden Spalten in einer gemeinsamen Darstellung wird Hamming-Gewicht der Darstellung genannt, und deren Dichte ist der Quotient des Hamming-Gewichts zur Länge m.

Das durchschnittliche Hamming-Gewicht einer gemeinsamen Darstellung zweier N[on]A[djacent]F[orm]s ist 5/9. Es ist möglich zu beweisen, dass die zu erwartende Anzahl von Multiplikationen in der Hauptschleife des vorstehenden zweiten Algorithmus aus dem dritten Ausführungsbeispiel $11n/27$ ist (vgl. R. Avanzi, "On the complexity of certain multi-exponentiation techniques in cryptography", erscheint in Journal of Cryptology), wobei die zusätzlichen Gruppenoperationen, die eventuell von der almost-online-Technik verursacht werden, ausgeschlossen sind.

Die Annahme, dass L entweder die native Wortlänge der zentralen Recheneinheit (= C[entral]P[rocessing]U[nit]) der Smart-Card oder ein kleines Vielfaches davon ist, zum Beispiel $L = 32$, ermöglicht auch eine einfachere Implementierung.

Mit Exponenten mit 160 Bits und unter Beachtung der Tatsache, dass eine N[on] A[djacent]F[orm] effizienterweise mit nur zwei Bits pro Koeffizientem gespeichert werden kann, werden etwa sechzehn Bytes R[andom]A[ccess]M[emory] für die Speicherung der zwei rekodierten 32 Bit-Brocken (= der zwei rekodierten 32 Bit-Abschnitte oder der zwei rekodierten 32 Bit-Teile) anstelle der achtzig Bytes für die vollen Exponenten benötigt. Die Speicherersparnis entspricht dem Speicherbedarf eines Punkts in projektiven Koordinaten auf einer elliptischen Kurve über einem endlichen Körper mit 160 Bits, ist also nicht so beträchtlich wie in den vorhergehenden beiden Ausführungsbeispielen.

Ausgehend von einem Computerprogramm, das die Anzahl der Fenster abzählt, die vom vorstehenden zweiten Algorithmus aus dem dritten Ausführungsbeispiel auf Zahlenpaaren gegebener Länge gebildet werden, kann dann der Durchschnitt der Resultate von einhunderttausend Durchläufen des Programms berechnet werden:

Die durchschnittliche Anzahl der Fenster auf Zahlenpaaren mit 160 Bits ist 65,81153 (zu beachten ist, dass $(11/27)\cdot 160 = 65{,}185$), die durchschnittliche Anzahl der Fenster auf Zahlenpaaren mit 32 Bits ist 13,64216 (zu beachten ist, dass $(11/27)\cdot 32 = 13{,}037$). Infolgedessen ist zu erwarten, falls $n = 160$ und $L = 32$, dass der almost-online-Algorithmus nur $5\cdot 13{,}64216 - 65{,}81153 = 2{,}39927$, das heißt etwa 2,4 mehr Gruppenoperationen als der vorstehende erste Algorithmus aus dem dritten Ausführungsbeispiel benötigt.

Da 235 die zu erwartende Gesamtanzahl der Gruppenoperationen des vorstehenden ersten Algorithmus aus dem dritten Ausführungsbeispiel im Fall $n = 160$ ist, kann abgeschätzt werden, dass der von der erfindungsgemäß eingesetzten almost-online-Technik verursachte Leistungsverlust etwa ein Prozent beträgt.

Es besteht eine alternative Darstellung zur N[on]A[djacent]F[orm] mit dem gleichen Hamming-Gewicht, die mit einem einfachen links-nach-rechts-arbeitenden (vgl. M.

Joye und S.-M. Yen, "Optimal left-to-right binary signed-digit recoding", IEEE Transactions on Computers 49 (7), 2000, Seiten 740 bis 748) Algorithmus berechnet werden kann. Mithin kann gefragt werden, ob diese Darstellung nicht anstelle der almost-online-Rekodierung benutzt werden könnte. Der Grund dafür, dass die Antwort negativ ausfällt, ist darin zu sehen, dass diese Alternative die N[on]A[djacent]F[orm]-Eigenschaft nicht aufweist, das heißt zwei aufeinanderfolgende Koeffizienten dürfen beide nicht verschwinden.

Die diesbezüglichen Auswirkungen auf den Speicherbedarf sind sehr schlecht. Im vorliegenden Falle $w = 2 = d$ würde die Menge E aus den Elementen $g_1^a g_2^b$ mit entweder $0 < a \leq 3$ und $-3 \leq b \leq 3$, wobei a und/oder b ungerade ist, oder $a = 0$ und $b = 1$ oder $b = 3$ bestehen; demzufolge hätte die Menge E die Mächtigkeit 20; dies würde den Speicherverbrauch des vorstehenden ersten Algorithmus aus dem dritten Ausführungsbeispiel zu groß machen.

Eine ähnliche Betrachtung trifft auf Solinas' "J[oint]S[parse]F[orm] - gemeinsame dünnbesetzte Darstellung" (vgl. J. A. Solinas, "Low-Weight Binary Representations for Pairs of Integers", Centre for Applied Cryptographic Research, University of Waterloo, Combinatorics and Optimization Research Report CORR 2001-41, 2001, erhältlich unter http://www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps) zu:

Die gemeinsame dünnbesetzte Darstellung rekodiert die beiden Exponenten gleichzeitig und voneinander abhängig. Die durchschnittliche Dichte der J[oint]S[parse]F[orm] ist 1/2 und die Anzahl der Gruppenoperationen in der Hauptschleife des vorstehenden ersten Algorithmus aus dem dritten Ausführungsbeispiel mit $w = 2 = d$ ist $3n/8$ (wie zuvor, ohne die Vorberechnungen und die Kosten von almost-online-Rekodierung hinzuzuziehen).

Die Anzahl vorberechneter Punkte beträgt zwölf, und das ist viel größer als die Anzahl acht der vorstehend vorgeschlagenen Variante, ohne dass der Durchsatz des Algorith-

mus mit Eingaben von 160 Bits bis 256 Bits beträchtlich besser wird. Für eine ausführlichere Diskussion sowie für entsprechende Beweise kann auf die Abschnitte 3.3 und 4.4 von H. Cohen, "Analysis of the flexible window powering algorithm", Vorabdruck erhältlich unter http://www.math.u-bordeaux.fr/~cohen/ verwiesen werden.

Viertes Ausführungsbeispiel: einfache Skalarmultiplikation

Die einfache Skalarmultiplikation in einer additiv geschriebenen Abelschen Gruppe G ergibt sich im Vergleich zum vorstehenden ersten Ausführungsbeispiel (einfache Exponenzierung = "single exponentiation") durch offensichtliche Ersetzungen [<--> neutrales Element "0", "Verdoppeln", "Summe" bei der Skalarmultiplikation anstelle von neutrales Element "1", "Quadrieren", "Produkt" bei der Potenzierung] und wird nachfolgend im Rahmen des vierten Ausführungsbeispiels der almost-online-Rekodierung als Algorithmus vorgestellt; bei dem

- ein Basiselement g der Abelschen Gruppe G,
- eine ganze Zahl e mit n Bits,
- eine Fensterbreite w und
- eine Brocken- oder Teilbreite $L >> w$

eingegeben werden; die (Einfach-)Skalarmultiplikation $e \cdot g$ wird ausgegeben:

Schritt 1 $x \leftarrow 0$
Schritt 2 $r \leftarrow \lceil n/L \rceil$, dann betrachte $e = \sum_{k=0}^{r-1} e_k 2^{kL}$ mit $0 \leq e_k < 2^L$.
Schritt 3 for $k = r-1$ downto 0 do {
(a) Rekodiere $(e_k) \rightarrow e_k = \sum_{j=0}^{L} b_j 2^j$
(b) if $b_L \neq 0$ then $x \leftarrow x + b_L g$
(c) for $j = L-1$ downto 0 do {
(i) $x \leftarrow 2x$
(ii) $x \leftarrow x + b_j g$. } }
Schritt 4 return x.

Hierzu ist analog zum ersten Ausführungsbeispiel anzumerken, dass es nach jeweils L Bits passieren kann, dass der vorstehende Algorithmus zwei Gruppenmultiplikationen in einer Reihe anstelle von nur einer Gruppenmultiplikation durchführt. Dies geschieht, wenn einer der Brocken e_i (= einer der Exponententeile e_i) eine ungerade Zahl darstellt

und wenn das Rekodieren des folgenden Brockens e_{i+1} (= des folgenden Exponententeils e_{i+1}) um einen Koeffizienten länger ist (b_L ungleich Null).

Fünftes Ausführungsbeispiel: Multiskalarmultiplikation

Der vorstehende Algorithmus aus dem vierten Ausführungsbeispiel (einfache Skalarmultiplikation) kann in ein Multi(skalar)multiplizierungsverfahren transformiert werden. Hierbei ergibt sich die Multiskalarmultiplikation in einer additiv geschriebenen Abelschen Gruppe G im Vergleich zum vorstehenden zweiten Ausführungsbeispiel (Multipotenzierung) durch offensichtliche Ersetzungen [<--> neutrales Element "0", "Verdoppeln", "Summe" bei der Multiskalarmultiplikation anstelle von neutrales Element "1", "Quadrieren", "Produkt" bei der Multipotenzierung] und wird nachfolgend im Rahmen des fünften Ausführungsbeispiels der almost-online-Rekodierung als Algorithmus vorgestellt.

Wenn Gruppenelemente $g_1, ..., g_d \in G$ und Exponenten $e_1, ..., e_d$ mit $d > 1$ gegeben sind und $\Sigma_{i=1}^{d} e_i \cdot g_i$ berechnet werden soll, wird zunächst eine Entscheidung für eine Verwendung einer dünnbesetzten Rekodierung der Exponenten $e_1, ..., e_d$ getroffen; dann wird eine "square-and-multiply"-Schleife verwendet:

Zuerst werden alle Vielfache $c \cdot g_i$ berechnet und gespeichert, wobei c ein zulässiger positiver Koeffizient ist. Dann wird eine temporäre Variable x auf $0 \in G$ gesetzt. Für $j = n, n-1, ..., 0$ wird zuerst x verdoppelt, und für $i = 1, ..., d$ wird zum verdoppelten x der Operand $e_{i,j} \cdot g_i$ hinzuaddiert, wobei $e_{i,j}$ der Koeffizient von 2^j in der Rekodierung von e_i ist. Am Ende enthält die temporäre Variable x das gewünschte Resultat.

Diese Methode wird auch als durchgeschobene Multiplikation bezeichnet; wie schon bei der Situation gemäß dem vierten Ausführungsbeispiel ist es wiederum wünschenswert, die Vorteile einer guten rechts-nach-links-Rekodierung beizubehalten, ohne zu viel Speicher verwenden zu müssen.

Die folgende Variante führt die Rekodierung "almost online", das heißt nahezu während der durchgeschobenen Multiskalarmultiplikation oder kurz nach der durchgeschobenen Multiskalarmultiplikation durch, wobei in den Algorithmus

- Basiselemente $g_1, ..., g_d$ der Abelschen Gruppe G,
- ganze Zahlen $e_1, ..., e_d$ ($d > 1$) mit jeweils höchstens n Bits,
- eine Fensterbreite w,
- eine Brocken- oder Teilbreite $L >> w$ und
- vorberechnete Vielfache $c \cdot g_i$ für alle c in der Koeffizientenmenge

eingegeben werden; das Multiskalarprodukt $\Sigma_{i=1}{}^{d}\, e_i \cdot g_i$ wird ausgegeben:

Schritt 1 $x \leftarrow 0$

Schritt 2 $r \leftarrow \lceil n/L \rceil$, dann betrachte $e_i = \sum_{k=0}^{r-1} e_{i,k} 2^{kL}$ mit $0 \leq e_{i,k} < 2^L$ und $i = 1, \ldots, d$.

Schritt 3 for $k = r-1$ downto 0 do {

(a) for $i = 1$ to d do {

Rekodiere $(e_{i,k}) \rightarrow e_{i,k} = \sum_{j=0}^{L} b_{i,j} 2^j$

if $b_{i,L} \neq 0$ then $x \leftarrow x + b_{i,L} g_i$ }

(b) for $j = L-1$ downto 0 do {

(i) $x \leftarrow 2x$

(ii) for i=1 to d do { if $b_{i,L} \neq 0$ then $x \leftarrow x + b_{i,j} \cdot g_i$.}

} }

Schritt 4 return x.

Als abschließender Teil der Beschreibung wird nachstehend eine Liste der vorliegend angeführten Anzahlen, Elemente, Exponenten, Gruppen, Indices, Koeffizienten, Mengen, Parameter, Skalare, Variablen und Zahlen gegeben:

$b_{i,j}$ Koeffizient

$b_{i,L}$ der höchsten Zweierpotenz 2^L zugeordneter Koeffizient

c zulässiger positiver Koeffizient

C endliche Menge ganzer Zahlen

d Anzahl der (Basis- oder Gruppen-)Elemente g_i aus der Gruppe G
= Anzahl der den (Basis- oder Gruppen-)Elementen g_i zugehörigen Exponenten oder Skalare e_i

e Exponent, insbesondere ganzzahliger Exponent, bei Einfachpotenzierung oder

	Skalar, insbesondere ganzzahliger Skalar, bei Einfachskalarmultiplikation
e_i	Exponent, insbesondere ganzzahliger Exponent, bei Multipotenzierung oder Skalar, insbesondere ganzzahliger Skalar, bei Multiskalarmultiplikation
$e_{i,k-1}$	dem (Exponenten- oder Skalar-)Brocken oder (Exponenten- oder Skalar-)Teil $e_{i,k}$ folgender (Exponenten- oder Skalar-)Brocken bzw. folgender (Exponenten- oder Skalar-)Teil
$e_{i,k}$	(Exponenten- oder Skalar-)Brocken oder (Exponenten- oder Skalar-)Teil des aufgeteilten Exponenten oder Skalars e_i
g	(Basis- oder Gruppen-)Element bei Einfachpotenzierung oder bei Einfachskalarmultiplikation
g_i	(Basis- oder Gruppen-)Element bei Multipotenzierung oder bei Multiskalarmultiplikation
G	Gruppe, insbesondere Abelsche Gruppe
i	Index
j	Index, insbesondere Summationsindex
k	Variable, insbesondere Laufvariable
L	(Exponenten- oder Skalar-)Brockenbreite oder (Exponenten- oder Skalar-)Teilbreite, insbesondere Bitanzahl der (Exponenten- oder Skalar-)Brockenbreite bzw. der (Exponenten- oder Skalar-)Teilbreite
n	maximale Bitanzahl oder maximale Bitlänge
r	Anzahl der (Exponenten- oder Skalar-)Brocken oder (Exponenten- oder Skalar-)Teile $e_{i,k}$
w	Parameter
x	temporäre Variable

PATENTANSPRÜCHE

1. Verfahren zur Multipotenzierung ($\Pi_{i=1}^{d} g_i^{e_i}$) bzw. zur Multiskalarmultiplikation ($\Sigma_{i=1}^{d} e_i \cdot g_i$) von Elementen ($g_i$) mittels jeweils mindestens eines jeweils eine maximale Bitanzahl (n) oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars (e_i), insbesondere zur Potenzierung (g^e) bzw. zur Skalarmultiplikation ($e \cdot g$) eines Elements (g) mittels mindestens eines eine maximale Bitanzahl (n) oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars (e), welche Elemente (g_i; g) mindestens einer

- im Falle der (Multi-)Potenzierung insbesondere multiplikativ bzw.
- im Falle der (Multi-)Skalarmultiplikation insbesondere additiv

notierten, zum Beispiel Abelschen Gruppe (G) entstammen,
gekennzeichnet durch
die folgenden Verfahrensschritte:

[a.1] Berechnen und Speichern oder

[a.2] Wiedererlangen aus mindestens einem Speicher

aller Potenzen (g_i^c) bzw. aller Vielfachen ($c \cdot g_i$), wobei c ein zulässiger positiver Koeffizient ist;

[b] Unterteilen eines jeden Exponenten bzw. Skalars (e_i) in mehrere Brocken oder in mehrere Teile ($e_{i,k}$) mit einer durch eine bestimmte Bitanzahl (L) gegebenen Brocken- oder Teilbreite; und

[c] einzelnes Rekodieren der Brocken oder Teile ($e_{i,k}$).

2. Verfahren gemäß Anspruch 1,
dadurch gekennzeichnet,
dass der Exponent bzw. Skalar (e_i) in der unterteilten Form $e_i = \Sigma_{k=0}^{r} e_{i,k} 2^{kL}$ dargestellt

wird, wobei

- r als die Anzahl der Brocken oder Teile ($e_{i,k}$), insbesondere als ganzzahliger Quotient aus der maximalen Bitanzahl (n) und der Bitanzahl (L) der Brocken- oder Teilbreite, definiert wird und
- $0 \leq e_{i,k} < 2^L$ ist.

3. Verfahren gemäß Anspruch 1 oder 2,
dadurch gekennzeichnet,
dass die Brocken- oder Teilbreite (L)

- wesentlich größer als ein Parameter (w), der der Breite, insbesondere der oberen Schranke der Breite, eines Fensters entspricht, über dem die Bits des jeweiligen Exponenten bzw. Skalars (e_i) gelesen werden, und
- wesentlich kürzer als die maximale Länge eines jeden Exponenten bzw. Skalars (e_i)

gewählt wird, insbesondere vor dem Verfahrensschritt [a.1] und/oder [a.2] gewählt wird.

4. Verfahren gemäß mindestens einem der Ansprüche 1 bis 3,
dadurch gekennzeichnet,

- dass im Falle der (Multi-)Potenzierung der Verfahrensschritt [c] des Rekodierens der Brocken oder Teile ($e_{i,k}$) für jeden einzelnen Brocken oder für jeden einzelnen Teil ($e_{i,k}$) eines jeden Exponenten (e_i) in folgende Teilschritte unterteilt werden kann:

[c.1] Setzen einer temporären Variable (x) auf einen normierten Wert, insbesondere auf den Wert 1 des in bezug auf die der Gruppe (G) zugeordnete Gruppenoperation neutralen Elements der Gruppe (G);

[c.2] sukzessives Setzen einer Variable (k) auf die Werte r-1, r-2, ..., 0, wobei für jeden Wert $k = r$-1, r-2, ..., 0 der Variable (k) die folgenden Teilschritte ausgeführt werden:

[c.2.i] für jeden Wert $i = 1, 2, ..., d$ eines Index (i), wobei d als die Anzahl der Elemente (g_i), insbesondere entsprechend der Anzahl der den Elementen (g_i) zugeordneten Exponenten (e_i), definiert wird:

[c.2.i.a] Rekodieren des Brockens oder Teils ($e_{i,k}$) als Summe ($\Sigma_{j=0}^{L} b_{i,j}2^j$) aus mit jeweils mindestens einem mindestens einer endlichen Menge (C) ganzer Zahlen entstammenden Koeffizienten ($b_{i,j}$) gewichteten Zweierpotenzen (2^j);

[c.2.i.b] wenn der der höchsten Zweierpotenz (2^L) zugeordnete Koeffizient ($b_{i,L}$) nicht verschwindet: Setzen der temporären Variable (x) auf das Produkt aus temporärer Variable (x) und der dem Koeffizienten ($b_{i,L}$) der höchsten Zweierpotenz (2^L) zugeordneten Potenz ($g_i^{b_{i,L}}$) des Elements (g_i);

[c.2.ii] für jeden Wert $j = L\text{-}1, L\text{-}2, ..., 0$ des Index (j):

[c.2.ii.a] Quadrieren der temporären Variable (x);

[c.2.ii.b] für jeden Wert $i = 1, 2, ..., d$ des Index (i):

wenn der der Zweierpotenz (2^j) zugeordnete Koeffizient ($b_{i,j}$) nicht verschwindet: Setzen der temporären Variable (x) auf das Produkt aus temporärer Variable (x) und der dem jeweiligen Koeffizienten ($b_{i,j}$) der Zweierpotenz (2^j) zugeordneten Potenz ($g_i^{b_{i,j}}$) des Elements (g_i); und

- dass nach dem Verfahrensschritt [c] des einzelnen Rekodierens der Brocken oder Teile ($e_{i,k}$) die temporäre Variable (x) wiedergegeben wird.

5. Verfahren gemäß mindestens einem der Ansprüche 1 bis 3,
dadurch gekennzeichnet,

- dass im Falle der (Multi-)Skalarmultiplikation der Verfahrensschritt [c] des Rekodierens der Brocken oder Teile ($e_{i,k}$) für jeden einzelnen Brocken oder für jeden einzelnen Teil ($e_{i,k}$) eines jeden Skalars (e_i) in folgende Teilschritte unterteilt werden kann:

[c.1] Setzen einer temporären Variable (x) auf einen normierten Wert, insbesondere

auf den Wert 0 des in bezug auf die der Gruppe (G) zugeordnete Gruppenoperation neutralen Elements der Gruppe (G);

[c.2] sukzessives Setzen einer Variable (k) auf die Werte r-1, r-2, ..., 0, wobei für jeden Wert $k = r$-1, r-2, ..., 0 der Variable (k) die folgenden Teilschritte ausgeführt werden:

[c.2.i] für jeden Wert $i = 1, 2, ..., d$ eines Index (i), wobei d als die Anzahl der Elemente (g_i), insbesondere entsprechend der Anzahl der den Elementen (g_i) zugeordneten Skalare (e_i), definiert wird:

[c.2.i.a] Rekodieren des Brockens oder Teils ($e_{i,k}$) als Summe ($\Sigma_{j=0}{}^{L} b_{i,j}2^j$) aus mit jeweils mindestens einem mindestens einer endlichen Menge (C) ganzer Zahlen entstammenden Koeffizienten ($b_{i,j}$) gewichteten Zweierpotenzen (2^j);

[c.2.i.b] wenn der der höchsten Zweierpotenz (2^L) zugeordnete Koeffizient ($b_{i,L}$) nicht verschwindet: Setzen der temporären Variable (x) auf die Summe aus temporärer Variable (x) und des dem Koeffizienten ($b_{i,L}$) der höchsten Zweierpotenz (2^L) zugeordneten Vielfachen ($b_{i,L}{\cdot}g_i$) des Elements (g_i);

[c.2.ii] für jeden Wert $j = L$-1, L-2, ..., 0 des Index (j):

[c.2.ii.a] Verdoppeln der temporären Variable (x);

[c.2.ii.b] für jeden Wert $i = 1, 2, ..., d$ des Index (i):

wenn der der Zweierpotenz (2^j) zugeordnete Koeffizient ($b_{i,j}$) nicht verschwindet: Setzen der temporären Variable (x) auf die Summe aus temporärer Variable (x) und des dem Koeffizienten ($b_{i,j}$) der Zweierpotenz (2^j) zugeordneten Vielfachen ($b_{i,j}{\cdot}g_i$) des Elements (g_i); und

- dass nach dem Verfahrensschritt [c] des einzelnen Rekodierens der Brocken oder Teile ($e_{i,k}$) die temporäre Variable (x) wiedergegeben wird.

6. Verfahren gemäß mindestens einem der Ansprüche 1 bis 5,
dadurch gekennzeichnet,

- dass der rekodierte Brocken oder der rekodierte Teil ($e_{i,k}$) einmal verwendet wird und
- dass die Speichereinheit, in der der rekodierte Brocken oder der rekodierte Teil ($e_{i,k}$) gespeichert wird, für das Rekodieren des folgenden Brockens oder des folgenden Teils ($e_{i,k-1}$) eingesetzt wird.

7. Verfahren gemäß mindestens einem der Ansprüche 1 bis 6,
dadurch gekennzeichnet,
dass das Verfahren auf mindestens einem insbesondere mindestens einer Chipkarte und/oder insbesondere mindestens einer Smart-Card zugeordneten Mikroprozessor implementiert wird.

8. Mikroprozessor, arbeitend gemäß einem Verfahren gemäß mindestens einem der Ansprüche 1 bis 7.

9. Vorrichtung, insbesondere Chipkarte und/oder insbesondere Smart-Card, aufweisend mindestens einen Mikroprozessor gemäß Anspruch 8.

10. Verwendung eines Verfahrens gemäß mindestens einem der Ansprüche 1 bis 7 und/oder mindestens eines Mikroprozessors gemäß Anspruch 8 und/oder mindestens einer Vorrichtung, insbesondere mindestens einer Chipkarte und/oder insbesondere mindestens einer Smart-Card, gemäß Anspruch 9 bei mindestens einem Kryptosystem, insbesondere bei mindestens einem Public-Key-Kryptosystem, bei mindestens einem Schlüsselaustauschsystem oder bei mindestens einem Signatursystem.

ZUSAMMENFASSUNG

Verfahren zum Potenzieren bzw. Multiplizieren von Elementen

Um ein Verfahren zur Multipotenzierung ($\Pi_{i=1}^{d} g_i^{e_i}$) bzw. zur Multiskalarmultiplikation ($\Sigma_{i=1}^{d} e_i \cdot g_i$) von Elementen ($g_i$) mittels jeweils mindestens eines jeweils eine maximale Bitanzahl (n) oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars (e_i), insbesondere zur Potenzierung (g^e) bzw. zur Skalarmultiplikation ($e \cdot g$) eines Elements (g) mittels mindestens eines eine maximale Bitanzahl (n) oder Bitlänge aufweisenden, insbesondere ganzzahligen Exponenten bzw. Skalars (e), welche Elemente (g_i; g) mindestens einer

- im Falle der (Multi-)Potenzierung insbesondere multiplikativ bzw.
- im Falle der (Multi-)Skalarmultiplikation insbesondere additiv

notierten, zum Beispiel Abelschen Gruppe (G) entstammen, so weiterzubilden, dass auch und gerade in extrem eingeschränkten Umgebungen, wie etwa bei Smart-Cards, der Bedarf an Speicherplatz für rekodierte Exponenten bzw. Skalare (e_i) so stark wie möglich verringert wird, werden die folgenden Verfahrensschritte vorgeschlagen:

[a.1] Berechnen und Speichern oder

[a.2] Wiedererlangen aus mindestens einem Speicher
aller Potenzen (g_i^c) bzw. aller Vielfachen ($c \cdot g_i$), wobei c ein zulässiger positiver Koeffizient ist;

[b] Unterteilen eines jeden Exponenten bzw. Skalars (e_i) in mehrere Brocken oder in mehrere Teile ($e_{i,k}$) mit einer durch eine bestimmte Bitanzahl (L) gegebenen Brocken- oder Teilbreite; und

[c] einzelnes Rekodieren der Brocken oder Teile ($e_{i,k}$).